UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/862,654	05/22/2001	Nigel Peter Topham	0808.65559	8723

24978 7590 06/28/2005

GREER, BURNS & CRAIN
300 S WACKER DR
25TH FLOOR
CHICAGO, IL 60606

EXAMINER	
LI, AIMEE J	

ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 06/28/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/862,654	TOPHAM, NIGEL PETER	
	Examiner	Art Unit	
	Aimee J. Li	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *13 January 2005 and 19 April 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *27-36 and 38-51* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *27-36 and 38-51* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper.No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 27-36 and 38-51 have been considered. Claims 1-13, 15-26, and 37 have been cancelled as per Applicant's request.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: RCE as received on 13 January 2005; Extension of Time 2 months as received on 13 January 2005; and Amendment as received on 19 April 2005.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 27-35 and 38-51 are rejected under 35 U.S.C. 103(a) as being unpatentable over Faraboschi et al., U.S. Patent Number 5,870,576.

5. Regarding claim 27, Faraboschi has taught a method of compressing a program to be executed by a processor in which compressed- form instructions stored in a program memory (110 of Fig.3, also see Col.4 lines 45-46) are decompressed and cached in an instruction cache prior to being issued (see Col.5 lines 29-32), the method comprising:

a. Converting a sequence of original instructions of the program into a corresponding sequence of compressed-form instructions (see Col.1 line 44 - Col.2 line 22),

b. Assigning such original instructions imaginary addresses according to said

sequence thereof, the assigned imaginary addresses being imaginary addresses at which the instructions are considered to exist when held in decompressed form in said instruction cache of the processor. Here, although not explicitly taught, it is inherent that during the compilation and storing of the program, the original instructions must be assigned addresses. As the decompressed form of the instructions are the same as the original form, the addresses will be the same.

6. Faraboschi has not explicitly taught wherein the method further comprises outputting a compressed program storable in said program memory and comprising the compressed-form instructions together with imaginary address information specifying said assigned imaginary address of at least one said original instruction so that, when the compressed- form instructions are decompressed and loaded by the processor into the instruction cache, the processor can allocate the assigned imaginary address to the decompressed instructions based on said imaginary address information.

7. However, "Official Notice" is taken that it is well known that a program counter stores address information that points to the address of an instruction to be accessed in the instruction memory (see lines in the code pointer segment having addresses in Fig.2) and that address information could easily be stored with the instruction to allow for faster lookups in the memory just like a cache tag. One of ordinary skill in the rot at the time of the invention would have recognized that by storing the address information along with the first one of the compressed sections in the memory (110 of Fig.2), it would allow for faster lookups and enable easy setting of the cache tag when loading the cache with instructions form the memory by copying the information into the cache tag. This information assigns the imaginary address of the

decompressed instruction because the program counter (200 of Fig.3) points to the cache block where the instruction is to be stored and the location in memory where the instruction information is stored Therefore, one of ordinary skill in the rot would have found it obvious to store the imaginary address information along with the section in the memory and use it to assign the cache tag and therefore the address of the cache block of the decompressed instruction, thereby allowing for faster lookups and shorter clock periods using simpler circuitry.

8. Regarding claim 28, Faraboschi has taught a method as claimed in claim 27, wherein the assigned imaginary addresses are selected so that instructions likely to coexist in the instruction cache at execution time will not be mapped to the same cache block (see Fig.2). Because the cache is direct mapped, the addresses assigned will not be mapped to the same cache block.

9. Regarding claim 29, Faraboschi has taught a method as claimed in claim 27, wherein the compressed-form instructions are arranged to be stored in said program memory in one or more compressed sections (see Fig.2), the compressed-form instructions belonging to each section occupying one cache block of the processor's instruction cache when decompressed (see Col.4 lines 48-67 and Cots lines 29-32), and at least one compressed section also containing imaginary address information relating to the instructions of that section (152 ofFig.2, imaginary address is stored in code pointer). Here, the program memory contains compressed-form instructions (WOO- W56 of Fig.2) stored in a "compressed section" of the heap (see address 14000300 in heap of Fig. 2), which are uncompressed and stored in a cache block of the cache (see WOO in row 040 of Fig.2).

10. Regarding claim 30, Faraboschi has taught a method as claimed in claim 29, but has not explicitly taught wherein said imaginary address information specifies the imaginary address at

which a first one of the decompressed instructions corresponding to said at least one compressed section is to be considered to exists when the decompressed instructions are held in said instruction cache.

11. However, "Official Notice" is taken that *it* is well known that a program counter stores address information that points to the address of an instruction to be accessed in the instruction memory (see lines in the code pointer segment having addresses in Fig.2) and that address information could easily be stored with the instruction to allow for faster lookups in the memory just like a cache tag. One of ordinary skill in the art at the time of the invention would have recognized that by storing the address information along with the compressed instructions in the memory (110 ofFig.2), it would allow for faster lookups and enable easy setting of the cache tag when loading the cache with instructions form the memory. This information assigns the imaginary address of the first one of the decompressed instruction because the program counter (200 ofFig.3) points to the cache block where the instruction is to be stored and the location in memory where the instruction information is stored. Therefore, one of ordinary skill in the art would have found it obvious to store the imaginary address information along with the section in the memory, thereby allowing for faster lookups and shorter clock periods using simpler circuitry.

12. Regarding claim 31, Faraboschi has taught a method as claimed in claim 29, but has not explicitly taught wherein said imaginary address information is contained in only a first one of said compressed sections to be loaded.

13. However, "Official Notice" is taken that it is well known that a program counter stores address information that points to the address of an instruction to be accessed in the instruction

memory (see Fig.2 where code pointer segments have addresses) and that address information could easily be stored with the instruction to allow for faster lookups in the memory just like a cache tag. However, in order to save space, only the location of the first entry of the section can be saved from which the other sections can be addressed using relative addressing. One of ordinary skill in the art would have recognized that by storing the address information along with the first one of the compressed sections in the memory, it would allow for faster lookups, savings in space as compared to storing the address information in all the sections, and enable easy setting of the cache tag when loading the cache with instructions from the memory. This information assigns the imaginary address of the decompressed instruction because the program counter (200 of Fig.3) points to the cache block where the instruction is to be stored and the location in memory where the instruction information is stored. Therefore, one of ordinary skill in the art would have found it obvious to store the imaginary address information along with the first one of the sections in the memory in order to allow for faster lookups and shorter clock periods using simpler circuitry.

14. Regarding claim 32, Faraboschi has taught a method as claimed in claim 29, wherein each said compressed section contains imaginary address information (152 of Fig.2) relating to the instructions belonging to the section concerned. Here, the code pointer contains imaginary address information relating to the compressed instructions belonging to the section concerned.

15. Regarding claim 33, Faraboschi has taught a method as claimed in claim 29, wherein the or each said compressed section further contains a decompression key (150 of Fig.2) for use by the processor to can)' out the decompression of the instructions belonging to said section (see Col.6 lines 29-34).

16. Regarding claim 34, Faraboschi has taught a method as claimed in claim 33, wherein said sequence of original instructions of the program comprises preselected instructions (NOP instructions indicated by the empty locations in cache of Fig.2) that are not stored explicitly in any said compressed section (see 140 ofFig.2), and the decompression key (150 of Fig.2) of the or each said compressed section identifies the positions at which said preselected instructions exist are to appear in a decompressed sequence of instructions corresponding to the section (see Col.5 lines 4-13). Here, the NOP instructions are not stored in the main (compressed) memory.

17. Regarding claim 35, Faraboschi has taught a method as claimed in claim 34, wherein said preselected instructions are "no operation" instructions (see Col.5 lines 4-13).

18. Regarding claim 38, Faraboschi has taught a processor, for executing instructions of a program stored in compressed form (see Col.4 lines 45-46, 58-67) in a program memory (see 110 of Fig.3), each said compressed-form instruction having an imaginary address at which the instruction is considered to exist when held in decompressed form within the processor (see Col.5lines 19-21), and imaginary address information from which the imaginary addresses assigned to the compressed-form instructions is derivable (see Fig.2, as well as Col.4 lines 5-8 and Col.5 lines 29-32), said processor comprising:

a. A program counter (132 ofFig.2), which identifies a position in said program memory (see Col.5lines 1-15 and Col.6lines 15-27),

b. An instruction cache (100 of Fig.3), having a plurality of cache blocks (see Fig.2), each for storing one or more instructions of said program in decompressed form (see Col.4 lines 5-8 and Col.5 lines 29-35),

c. An imaginary address deriving unit operable to derive therefrom the imaginary

address of at least a first one of the compressed-form instructions in said program (see Fig.2, as well as Col.4 lines 5-8 and Col.5 lines 29-32),

d. A cache loading unit (204/210/212 ofFig.3), comprising a decompression section (210/212 ofFig.3), operable to perform a cache loading operation in which one or more compressed-form instructions are read from said position in the program memory identified by the program counter and are decompressed and stored in one of said cache blocks of the instruction cache (see Col.5 lines 24-34 and Col.6 lines 11-53), which cache block is determined by the imaginary address of said one or more compressed-form instructions being read from said position in the program memory (see Fig.2, as well as Col.4 line 48 - Col.5line 21),

e. A cache pointer (200 ofFig.3), which identifies a position in said instruction cache of an instruction to be fetched for execution (see Col.5 lines 19-21, 55-57),

f. An instruction fetching unit (208/220 ofFig.3) which fetches an instruction to be executed from the position identified by the cache pointer (see Col.51ines 36-40) and which, when a cache miss occurs because the instruction to be fetched is not present in the instruction cache, causes the cache loading unit to perform said cache loading operation (see Col.5 lines 40-50),

g. An updating unit (206 ofFig.3) which updates the program counter (132 of Fig.2) and cache pointer (200 of Fig. 3) in response to the fetching of instructions so as to ensure that said position identified by said program counter is maintained consistently at the position in said program memory at which the instruction to be fetched form the *instruction* cache is stored in compressed form (see Col.5 lines

19-24 and Col.6 lines 11-53). Here, the program counter (132 ofFig.2), which is comprised of a mask (150 of *Fig.2)* and an offset (152 ofFig.2), are updated to point to the current position in the program memory when performing an instruction fetch upon a cache miss (see Col.6 lines 11-53). Thus, along with the updating of the cache pointer (see Col.5lines 19-24), the position of the current instruction to be fetched from the instruction cache is maintained. Further, on a cache miss, the instruction address in the program counter (200 of Fig.3) is used to access the code pointer segment (130 ofFig.2) in program memory (see Col.6lines 11-16). The code pointer (152 of Fig.2) is then used to locate the position in the program memory of the next compressed section following the compressed section corresponding to the most-recently-accessed cache block (see Col.61ines 16-24). The position located in the compressed memory is considered the next compressed section because the compressed sections are arranged in consecutive memory locations (see CoJ.6 lines 20-24).

19. Faraboschi has not explicitly taught wherein the program memory stores imaginary address information or an imaginary address deriving unit operable to read the imaginary address information stored in the program memory.

20. However, "Official Notice" is taken that it is well known that a program counter stores address information that points to the address of an instruction to be accessed in the instruction memory (see Fig.2 where code pointer segments have addresses) and that address information could easily be stored with the instruction to allow for faster lookups in the memory just like a cache tag. One of ordinary skill in the art would have recognized that by storing the address

information along with the compressed instructions in the memory (110 of Fig.2), it would allow for faster lookups and enable easy setting of the cache tag when loading the cache with instructions from the memory. Therefore, one of ordinary skill in the art would have found it obvious to store the imaginary address information along with the compressed instruction, thus allowing for faster lookups and shorter clock periods using simpler circuitry.

21. Regarding claim 39, Faraboschi has taught a processor as claimed in claim 38, wherein:

a. The compressed-form instructions are stored in the program memory in one or more compressed sections (see Fig.2), the compressed-form instructions belonging to each section occupying one of said cache blocks when decompressed (see Col.41ines 48-67 and Co1.5 lines 29-32), and at least one section also contains imaginary address information relating to the instructions belonging to the section (see Fig.2). Here, the imaginary address information is stored in the code pointer (152 of Fig.2), and the program memory contains compressed-form instructions (WOO- W56 of Fig.2) stored in a "compressed section" of the heap (see address 14000300 in heap of Fig.2), which are uncompressed and stored in a cache block of the cache (see WOO in row 040 ofFig.2).

b. Said cache loading unit is operable, in said cache loading operation, to decompress and load into one of said cache blocks one such compressed section stored at the position in the program memory identified by the program counter (see Col.5 lines 24-34 and Co1.6 lines 11-53).

22. Regarding claim 40, Faraboschi has taught a processor as claimed in claim 39, but has

not explicitly taught wherein said imaginary address information of said at least one section specifies the imaginary address at which a first one of the decompressed instructions corresponding to the compressed section is considered to exist when the decompressed instructions are held in one of the cache blocks.

23. However, "Official Notice" is taken that it is well known that a program counter stores address information that points to the address of an instruction to be accessed in the instruction memory (see Fig.2 where code pointer segments have addresses) and that address information could easily be stored with the instruction to allow for faster lookups in the memory just like a cache tag. One of ordinary skill in the art would have recognized that by storing the address information along with the compressed instructions in the memory (110 of Fig.2), it would allow for faster lookups and enable easy setting of the cache tag when loading instructions from the memory. This information would specify the imaginary address of the first one of the decompressed instructions because the program counter (200 of Fig.3) points to the cache block where the instruction is to be stored and the location in memory where the instruction information is stored. Therefore, one of ordinary skill in the art would have found it obvious to store the imaginary address information along with the section in memory in order to provide for faster lookups and shorter clock periods using simpler circuitry.

24. Regarding claim 41, Faraboschi has taught a processor as claimed in claim 39, wherein said imaginary address information is contained in only a first one of said compressed sections to be loaded (see Fig.2 and Col.4 lines 48-67 and Col.5 lines 29-32). Here, the imaginary address information is the mask (150 of Fig. 2) and the code pointer (152 of Fig.2), and the mask for a section is only stored in association with the first compressed section (see "mask 132" ofFig.2

for example).

25. Regarding claim 42, Faraboschi has taught a processor as claimed in claim 39, wherein each said compressed section contains imaginary address information relating to the instructions belonging to the section concerned (see Fig.2 and Col.4 lines 48-67 and CoI.5 lines 29-32). Here, the imaginary address information is the mask (150 of Fig.2) and the code pointer (152 of Fig.2), and the mask for a section is only stored in association with the first compressed section (see "mask 132" of Fig.2 for example).

26. Regarding claim 43, Faraboschi has taught a computer-readable recording medium storing a compressed program, said compressed program being adapted to be stored in a program memory of a processor (see *Col.4* lines 45-56) and comprising:

a. A sequence of compressed-form instructions derived from a corresponding sequence of original instructions (see Col.1 line 44 - Col.2 line 22), the compressed-form instructions being adapted to be decompressed by the processor (see Col.4 lines 45-56) and cached in an instruction cache thereof prior to issuance (see Co1.5lines 29-32),

b. Imaginary address information specifying an imaginary address assigned to at least one of said original instructions, being an imaginary address at which the original instruction is to be considered to exist when held in decompressed form in said instruction cache, whereby when the compressed-form instructions are decompressed (see Col.4 lines 45-56) and loaded by the processor into the instruction cache (see Col.5 lines 29-32). Here, although not explicitly taught, it is inherent that during the compilation and storing of the program, the original

instructions must be assigned addresses. Thus, as the decompressed forms of the instructions are the same as the original form, the addresses will be the same.

27. Faraboschi has not explicitly taught wherein the processor can allocate the decompressed instructions such imaginary addresses based on said imaginary address in formation.

28. However, "Official Notice" is taken that it is well known that a program counter stores address information that points to the address of an instruction to be accessed in the instruction memory (see Fig.2 where code pointer segments have addresses) and that address information could easily be stored with the instruction to allow for faster lookups in the memory just like a cache tag. One of ordinary skill in the art would have recognized that by storing the address information along with the compressed instructions in the memory (110 of Fig.2), it would allow for faster lookups and enable easy setting of the cache tag when loading instructions from the memory. This information would specify the imaginary address of the first one of the decompressed instructions because the program counter (200 of Fig.3) points to the cache block where the instruction is to be stored and the location in memory where the instruction information is stored. Therefore, one of ordinary skill in the art would have found it obvious to store the imaginary address information along with the section in memory in order to provide for faster lookups and shorter clock periods using simpler circuitry.

29. Regarding claim 44, Faraboschi has taught a computer-readable recording medium as claimed in claim 43, wherein the assigned imaginary addresses are selected so that instructions likely to coexist in the instruction cache at execution time will not be mapped to the same cache block (see Fig.2). Because the cache is direct mapped, the addresses assigned will not be mapped to the same cache block.

30. Regarding claim 45, Faraboschi has taught a computer-readable recording medium as claimed in claim 43, wherein the compressed-form instructions are arranged to be stored in the said program memory in one or more compressed sections (see Fig.2), the compressed-form instructions belonging to each section occupying one cache block of the processor's instruction cache when decompressed (see Col.4 lines 48-67 and Col.5 lines 29-32), and at least one compressed section also containing imaginary address information relating to the instructions of that section (152 of Fig.2, imaginary address is stored in code pointer). Here, the program memory contains compressed-form instructions (WOO-W56 ofFig.2) stored in a "compressed section" of the heap (see address 14000300 in heap of Fig.2), which are uncompressed and stored in a cache block of the cache (see WOO in row 0400fFig.2).

31. Regarding claim 46, Faraboschi has taught a computer-readable recording medium as claimed in claim 45, but has not explicitly taught wherein said imaginary address information specifies the imaginary address at which a first one of the decompressed instructions corresponding to said one compressed section is to be considered to exist when the decompressed instructions are held in the same instruction cache.

32. However, "Official Notice" is taken that it is well known that a program counter stores address information that points to the address of an instruction to be accessed in the instruction memory (see Fig.2 where code pointer segments have addresses) and that address information could easily be stored with the instruction to allow for faster lookups in the memory just like a cache tag. One of ordinary skill in the art would have recognized that by storing the address information along with the compressed instructions in the memory (110 of Fig.2), it would allow for faster lookups and enable easy setting of the cache tag when loading instructions from the

memory. This information would specify the imaginary address of the first one of the decompressed instructions because the program counter (200 of Fig.3) points to the cache block where the instruction is to be stored and the location in memory where the instruction information is stored. Therefore, one of ordinary skill in the art would have found it obvious to store the imaginary address information along with the section in memory in order to provide for faster lookups and shorter clock periods using simpler circuitry.

33. Regarding claim 47, Faraboschi has taught a computer-readable recording medium as claimed in claim 45, but has not explicitly taught wherein said imaginary address information is contained in only a first one of the said compressed sections to be loaded.

34. However, "Official Notice" is taken that it is well known that a program counter stores address information that points to the address of an instruction to be accessed in the instruction memory (see Fig.2 where code pointer segments have addresses) and that address information could easily be stored with the instruction to allow for faster lookups in the memory just like a cache tag. One of ordinary skill in the art would have recognized that by storing the address information along with the compressed instructions in the memory (110 of Fig.2), it would allow for faster lookups and enable easy setting of the cache tag when loading instructions from the memory. This information would specify the imaginary address of the first one of the decompressed instructions because the program counter (200 of Fig.3) points to the cache block where the instruction is to be stored and the location in memory where the instruction information is stored. Therefore, one of ordinary skill in the art would have found it obvious to store the imaginary address information along with the section in memory in order to provide for faster lookups and shorter clock periods using simpler circuitry.

35. Regarding claim 48, Faraboschi has taught a computer-readable recording medium as claimed in claim 45, wherein each said compressed section contains imaginary address information (152 ofFig.2) relating to the instructions belonging to the section concerned. Here, the code pointer contains imaginary address information relating to the compressed instructions belonging to the section concerned.

36. Regarding claim 49, Faraboschi has taught a computer-readable recording medium as claimed in claim 45, wherein the or each said compressed section further contains a decompression key (150 ofFig.2) for use by the processor to carry out the decompression of the instructions belonging to the said section (see Col.6 lines 29-34).

37. Regarding claim 50, Faraboschi has taught a computer-readable recording medium as claimed in claim 49, wherein said corresponding sequence of original instructions includes preselected instructions (NOP instructions indicated by the empty locations in cache of Fig. 2) that are not stored explicitly in any said compressed section (see 140 ofFig.2), and the decompression key of the or each said compressed section identifies the positions at which said preselected instructions exist are to appear in a decompressed sequence of instructions corresponding to the section (see Col.S lines 4-13). Here, the NOP instructions are not stored in the main (compressed) memory.

38. Regarding claim 51, Faraboschi has taught a computer-readable recording medium as claimed in claim 50, wherein said preselected instructions are "no operation" instructions (see Col.5 lines 4-13)

39. Claim 36 is rejected under 35 U.S.C. 103(a) as being unpatentable over Faraboschi et al., U.S. Patent No. 5,870,576, in view of Tannenbaum, *Structured Computer Organization.*

40. Regarding claim 36, Faraboschi has taught a computer-readable recording medium storing a computer program which carries out a method of compressing a processor program to be executed by a process, the processor being operable to decompress compressed-form instructions stored in a program memory (110 of Fig.3, also see Col.4 lines 45-46) and to cache the decompressed instructions in an instruction cache prior to issuing them (see CoL5 lines 29-32), the computer program comprising:

a. A converting portion which converts a sequence of original instructions of the processor program into a corresponding sequence of such compressed-form instructions (see Col.1 line 44 - Col2 line 22),

b. An assigning portion which assigns such original instructions imaginary addresses according to said sequence thereof, the assigned imaginary addresses being imaginary address at which the instructions are to be considered to exist when held in decompressed form in said instruction cache of the processor. Here, although not explicitly taught, it is inherent that during the compilation and storing of the program, the original instructions must be assigned addresses. As the decompressed forms of the instructions are the same as the original form, the addresses will be the same.

41. Faraboschi has not explicitly taught wherein the method further comprises an outputting portion which outputs a compressed program storable in said program memory and comprising the compressed-form instructions together with imaginary address information specifying said assigned imaginary address of at least one said original instruction so that, when the compressed-form instructions are decompressed and loaded by the processor into the instruction cache, the

processor can allocate the assigned imaginary address to the decompressed instructions based on said imaginary address information: Also, Faraboschi has not taught wherein the instructions on a computer-readable medium carry out the method of the invention as claimed.

42. However, "Official Notice" is taken that it is well known that a program counter stores address information that points to the address of an instruction to be accessed in the instruction memory (see Fig.2 where code pointer segments have addresses) and that address information could easily be stored with the instruction to allow for faster lookups in the memory just like a cache tag. One of ordinary skill in the art would have recognized that by storing the address information along with the compressed instructions in the memory (110 of Fig.2), it would allow for faster lookups and enable easy setting of the cache tag when loading instructions from the memory. This information would specify the imaginary address of the first one of the decompressed instructions because the program counter (200 of Fig.3) points to the cache block where the instruction is to be stored and the location in memory where the instruction information is stored. Therefore, one of ordinary skill in the art would have found it obvious to store the imaginary address information along with the section in memory in order to provide for faster lookups and shorter clock periods using simpler circuitry.

43. Furthermore, Tannenbaum has taught that any instruction executed by hardware can also be simulated in software (see Tannenbaum, p.ll, para. 4, lines 1-2). He also has taught that hardware is generally immutable (see first para. after sec. 1.4 header), while software allows for more rapid change (see Tannenbaum, p.ll, para. 4, lines 1-2). One of ordinary skill in the at.1 at the time of the invention would have been motivated to convert Faraboschi to software, i.e. instructions on a machine-readable medium because Tannenbaum has taught that hardware is

generally immutable while software allows for more rapid changes. Therefore, one of ordinary skill in the art would have found it obvious to modify Faraboschi to be instructions recorded on a machine readable medium in the manner suggested by Tannenbaum in order to allow for ease of correction of mistakes and/or an ease of addition of new functionality.

Response to Arguments

44. Applicant's arguments filed 13 January 2005 and 19 April 2005 have been fully considered but they are not persuasive. Applicant argues in essence on pages 12-14

> ...the examiner takes 'Official Notice' that it is well known that a computer counter stores address information that points to the address of an instruction to be accessed in the instruction memory, and their address information could be easily stored with the instruction to allow for faster look-ups in the memory just like a cache tag, but provides no evidence to support Official Notice of these multiple facts...

45. This has not been found persuasive. The examiner points Fabaroschi's Figure 2 to show the association of addresses with operations via the code pointer segment and its mask. The examiner provides Yoshida, U.S. Patent Number 5,799,138; Franaszek, U.S. Patent Number 5,864,859; and Kochar et al., U.S. Patent Number 6,658,548, which all specifically deal with compressed instructions and data. Each of the patents teaches that the addresses associated with an instruction and/or data are stored in a table or similar storage together. One specific example to note is in Yoshida in column 6, lines 55-65 where the virtual address of the code is stored with the address portion.

Conclusion

46. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Aimee J. Li whose telephone number is (571) 272-4169. The examiner can normally be reached on M-T 7:30am-5:00pm.

47. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

48. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

AJL
Aimee J. Li
23 June 2005

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100